JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION: RESEARCH AND PRACTICE
J. Softw. Maint. Evol.: Res. Pract. 2001; **13**:77–95 (DOI: 10.1002/smr.223)

Practice

MANTOOL: a tool for supporting the software maintenance process

Macario Polo Usaola*,†, Mario Piattini Velthuis and Francisco Ruiz González

Grupo Alarcos-Departament of Computer Science, University of Castilla-La Mancha, Ronda de Calatrava, 5, 13071-Ciudad Real, Spain

SUMMARY

In this paper we present MANTOOL, an automatic tool for managing the software maintenance process according to MANTEMA, a rigorous methodology for maintenance. After explaining briefly the MANTEMA structure, the article explains how MANTOOL allows users to manage modification requests following the different stages of the methodology, showing it with some examples. The data saved in MANTOOL can be used to extract different kinds of reports and to do estimations of future maintenance interventions. The paper also includes some experience reports about the construction and application of MANTEMA and MANTOOL, and some reflections on the benefits of its use. Copyright © 2001 John Wiley & Sons, Ltd.

KEY WORDS: software maintenance; maintenance tools; maintenance methodologies; process management

1. INTRODUCTION

Over the past few decades, costs of software maintenance have increased and the emerging current technologies suggest that this situation will continue for some years because of:

- the integration of e-commerce services into legacy applications [1];
- the maintenance of hypertextual documents [2]; and
- the adaptation of old applications to the object-oriented paradigm [3] or to other environments such as client/server or distributed computation [4].

*Correspondence to: M. Polo, Grupo Alarcos-Departament of Computer Science, University of Castilla- La Mancha, Ronda de Calatrava, 5, 13071-Ciudad Real, Spain.
†E-mail: mpolo@inf-cr.uclm.es

Contract/grant sponsor: CICYT/EU; contract/grant number: 1FD97-1608
Contract/grant sponsor: Ministerio de Industria y Energía, Iniciativa ATYCA; contract/grant number: TA15/1999

Received 31 January 2000
Revised 9 November 2000

However, most software organizations have their structures, methodologies and work teams prepared for new development, although the main source of work they do is maintenance. Singer [5] reveals that 61% of the professional life of programmers is devoted to maintenance work and only 39% to new development. Van Bon [6] remarks that the managing of the software maintenance process is still being neglected and that there are few specific methodologies for it. For Basili *et al*. [7], the definition and validation of methodologies that take into account the specific characteristics of the software maintenance organizations and of their processes are required. Nor do methodologies for software development consider the future stage of maintenance [8]. This fact is aggravated by situations like those described by Alexander and Davis [9], for whom the project managers usually choose a software process model *ad hoc*, unjustified and undocumented.

As is well known, 'a good methodology should lend itself to automation' [10] but, as Pigoski [11] states, there is a lack of definition of the 'maintenance process' which voids the implementation and use of CASE tools for its managing. On the other side, both the possession of a methodology and the use of tools are some of the control goals in audits of the 'maintenance process' [12].

According to Quang [13], there are many tools suitable for maintenance, although they were initially developed with other goals. This possibility is due to the similarities between some tasks of maintenance and other stages (specially design, coding and tests). However, although some of these tools are usable throughout the process (they can be considered as 'horizontal' tools), they cover only some of its 'bands', as for example, configuration management. As a matter of fact, the CASE tool index of the Queen's University [14] shows many maintenance tools, but almost all for reengineering and reverse engineering, and none for managing the process. Mazza *et al*. [15] presents the following taxonomy of CASE tools for maintenance:

- browsing tools, to easily locate the pieces of code we want to modify;
- code-improvement tools, for reformatting and restructuring the source code; and
- reverse-engineering tools, which process the source code and produce another type of software product (or, more generally, produce a software artifact at a higher abstraction level).

Tools in this taxonomy are 'vertical' tools, in the sense of that they help only in certain phases of the software maintenance process.

Therefore, there are two further important aspects. These are the definition and validation of a methodology for managing the software maintenance process and the development of a 'horizontal' tool for supporting such methodology. Over the last few years, we have proposed and refined MANTEMA, a methodology for managing the Software Maintenance Process [16] which is based on the ISO/IEC 12207 (ISO/IEC, 1995) Maintenance Process [17]. MANTEMA has been jointly developed by our research group and Atos ODS, an international organization among whose primary business activities is the outsourcing of software maintenance. Atos ODS is using MANTEMA in some of their projects in large banking entities. Nevertheless, the use of this methodology without an automatic tool which supports it was evidenced as its main drawback by managers and programmers, as coincides with Gillies's idea [10].

According to Graham, Henderson-Sellers and Younessi [18], some of the elements that a methodology must provide are:

- a process model;
- a set of techniques;

- a set of deliverables;
- a set of metrics;
- guides for the project management, including roles and team structure; and
- automatic tools for supporting the methodology.

In this sense, MANTEMA is a full methodology, since we have different kinds of results in each one of these six 'dimensions'. In this paper we briefly present the structure of the process model defined in the MANTEMA methodology and then we focus on how MANTOOL helps to manage the software maintenance process according to it.

The paper is organized as follows: in Section 2, we briefly set out the structure of MANTEMA and Section 3 is a presentation of MANTOOL, showing some examples of its application to real cases. Also in this section, a discussion of the benefits reported by its use is presented. In Section 4 we provide some experience reports on MANTEMA and MANTOOL. In Section 5, we draw our conclusions and state our future lines of work.

2. STRUCTURE OF MANTEMA

MANTEMA has been defined taking the maintenance process defined in ISO/IEC 12207 as a basis. This International Standard has been selected because, as Pigoski [11] says, it 'will drive the world software trade and will impact on maintenance'. Moreover, the construction and use of methodologies from this kind of standard may help to obtain certifications, such as ISO 9000. This aspect is very important for organizations such as our partner. Software maintenance services via outsourcing is a fast growing business activity [19–22].

The methodology can be seen as the multi-stage graph shown in Figure 1, whose initial node is devoted to the establishment of the outsourcing relationship (if appropriate) and to the preparation of the maintenance process. Then, the process enters into a cycled, iterative set of activities and tasks, in which the arrival of every modification request (MR) is done, in such manner that they are classified into one of the five types of maintenance defined in MANTEMA.

(i) *Urgent corrective*: when a detected error prevents normal system operation and the solution time is critical.
(ii) *Non-urgent corrective*: when a detected error does not block the normal operation of the system and the solution time is not critical.
(iii) *Perfective*: when new functionalities are added to the system.
(iv) *Preventive*: when software modification is done to improve its maintainability and quality properties.
(v) *Adaptive*: when the system will change its execution environment.

Every type has a different series of activities and tasks, although the last four types can be conceptually grouped into just one (the 'plannable' maintenance), since they share the same design guidelines and the practical application of the methodology has shown they have similar manners of execution. When the MR has been serviced, the migration and retirement of the software is executed following these activities, as imported from ISO/IEC 12207. When the maintenance process arrives at its end, the activities included in the final node of the graph must be realized. These include the end of intervention,

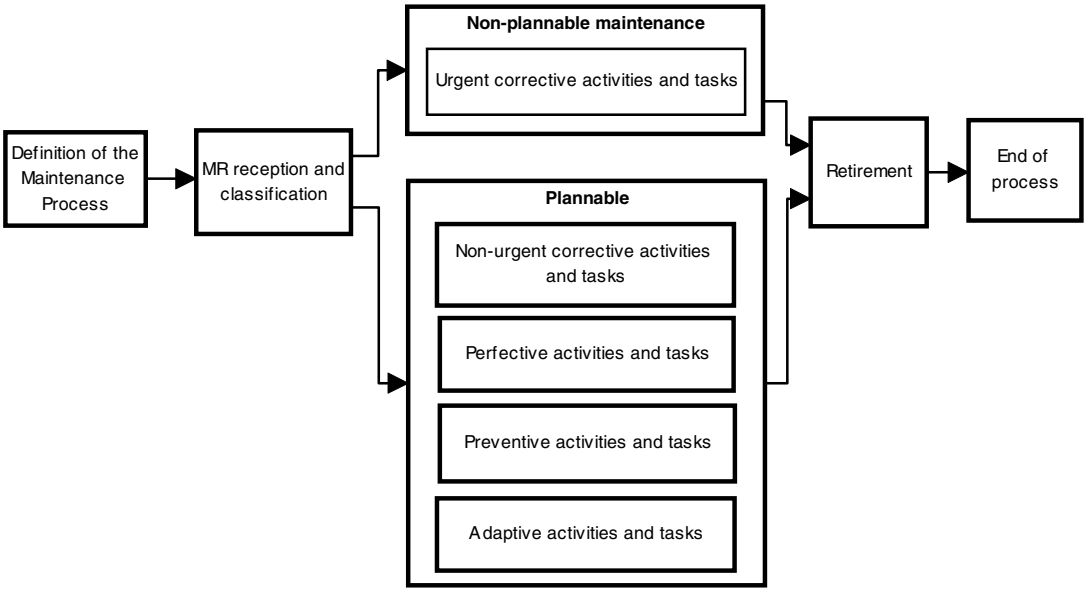

Figure 1. General structure of MANTEMA methodology.

the updating of the historic database with the collected data of metrics, the retirement (according to ISO/IEC 12207) and, if there has been outsourcing relationship, its finalization. In this last case, the maintenance supplier returns the inventory and documentation to the customer, educates the new maintenance personnel and definitively ends the service supply.

Every node of the methodology graph consists of a set of activities, which are composed of a set of tasks. For every task, MANTEMA determines input and output products, metrics to be collected, mentions suitable techniques to be used, suggests other life cycle processes to be used in the task (such as quality assurance, configuration management and training) and identifies the people responsible for the task's execution [23]. In this manner, every node may be represented using a table such as the one shown in Table I.

Among the proposed metrics for tasks, we recommend in MANTEMA the collection of different product metrics for programs (lines of code added/deleted/modified, cyclomatic complexity of routines before and after interventions etc.). However, we have also defined a wide set of metrics for different kinds of databases, due to their great influence on the complexity of current information systems and due to the lack of research in this field [24]. In this manner, we have defined and validated metrics for conceptual schemes [25], relational databases [26], object-relational databases [27] and active databases [28].

Setting aside usual and suitable techniques for maintenance, MANTEMA recommends the use of two novel techniques for maintenance management. The first is for estimating the quantity of resources

Table I. Example of the structure of every node: the urgent corrective maintenance.

Activity	Task	Inputs	Outputs	Techniques	Responsible	Interfaces with other processes
Error analysis	UC1 Investigating and analyzing causes	Software product in operation Urgent error Modification request	Software to be corrected		Maintenance team	Configuration Management
Corrective intervention	UC2 Making corrective actions	Software to be corrected	Corrected software	Codification DB design	Maintenance team	Quality Assurance
	UC3 Filling in documentation	Old software (with errors) New software (without visible errors)	Documentation with the corrective actions made		Maintenance team	
	UC4 Executing unitary tests	Corrected software Unitary test cases Integration test cases	Assurance of the correction of the error	Non-regression techniques	Maintenance team	Verification Validation
	UC5 Putting the software product into production environment	Corrected software	Corrected software in operation		Maintenance team	Configuration Management
Intervention close	UC6 Documenting correction	Documentation of: • Corrective actions • Tests	Full document		Maintenance team	
	UC7 Saving intervention	Full document	Stored full document		Maintenance team	Configuration Management

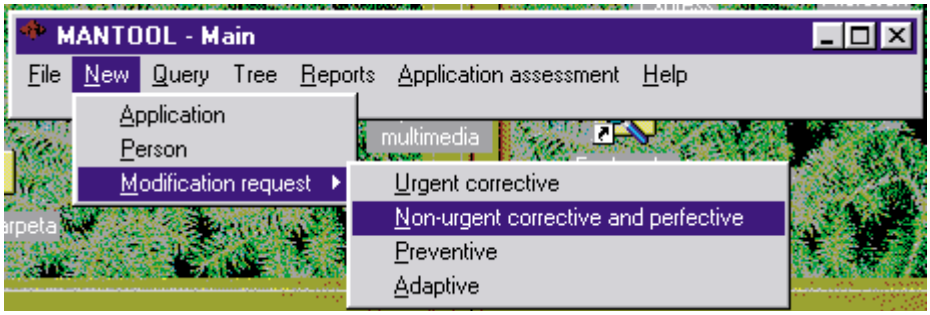

Figure 2. Main screen of MANTOOL.

to be devoted to corrective maintenance with no economic loss [29], and the second is for the early identifying and estimating of the risks associated with maintenance projects. This technique is very useful for the maintenance supplier before the acceptance of a maintenance contract.

3. MANTOOL

MANTOOL is a tool for managing the software maintenance process according to the MANTEMA methodology. As the reader has already seen, five types of maintenance are distinguished in the methodology. These are: *urgent corrective*, *non-urgent corrective*, *perfective*, *preventive* and *adaptive*. Tasks composing every one of these types are defined in a very precise way.

MANTOOL incorporates the managing of MRs from their inception until the end of their respective sets of activities and tasks, according to maintenance type. Figure 2 shows the main screen of MANTOOL with the menu for introducing a new MR unfolded.

Once the desired option has been selected, one screen for introducing the most relevant information of the MR is shown, as in Figure 3. We have illustrated this case with an *urgent corrective* MR related to an error in the execution of MANTOOL. Templates of MR and of other documents generated during the maintenance process are carefully detailed in MANTEMA [16].

The pursuit of every MR is done in a screen such as that shown in Figure 4. In this screen there exists a tab for every task belonging to the maintenance type of the MR; inputs, outputs and metrics of the task are specified in this screen. There is also an additional tab (the one in the foreground in Figure 4) which shows the general information about the MR. This comprises the application, date and time of presentation, the number of the MR, the last executed task, the user who presented it, and a description of the error and the error messages. There is also a graph with nodes associated with every iteration. Nodes change their colour when their respective tasks are executed in order to provide a quick idea of the tasks.

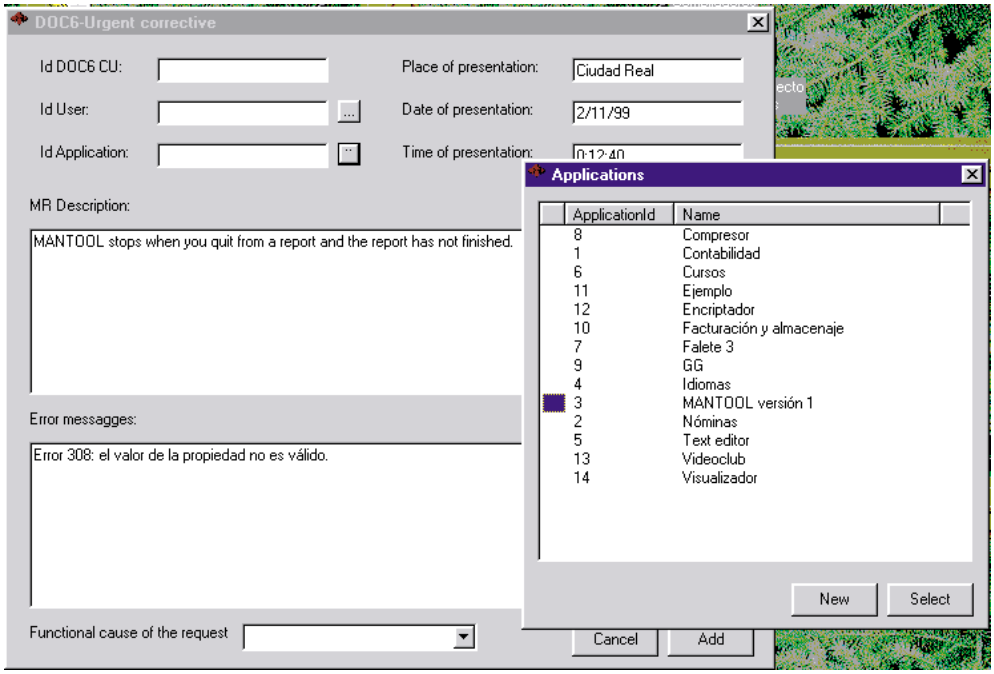

Figure 3. Screen for introducing *urgent corrective* MRs.

3.1. Processing a modification request

In this section, we will see how to use MANTOOL for executing the tasks corresponding to the *urgent corrective* MR number 71, whose information was introduced in Figure 3. We have selected an *urgent corrective* MR because this is the type of maintenance with fewest tasks and is the easiest to visualize.

3.1.1. Task UC1: investigating and analyzing causes

In this first task, a list of software elements to be corrected must be provided as an output (see Table I). The corresponding application to this MR was specified in Figure 3, through its selection from a list within the application portfolio. Using the button *See software elements*, the window highlighted in the upper part of Figure 5 appears. This window shows all the modules and routines of the application being maintained, along with the values of some metrics .

After having examined the code for detecting the error, the software elements to be corrected can be selected from this screen and passed to the text box labeled *Elements to be corrected*, where the user can add some comments. The 'software elements dictionary' can be filled automatically for Visual Basic

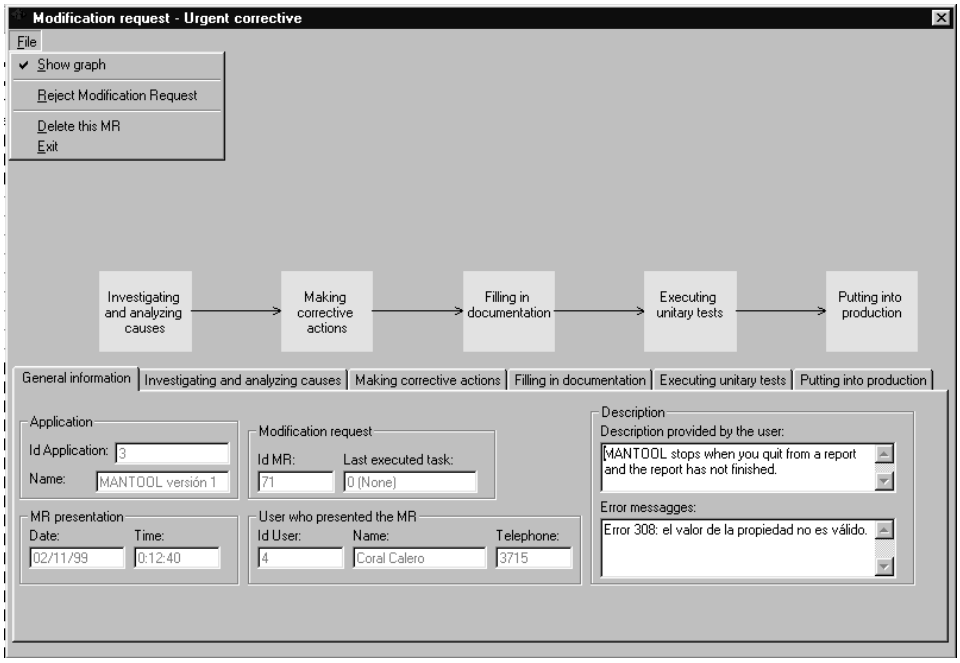

Figure 4. Status of a MR.

applications and Cobol programs (with two components incorporated in MANTOOL), or manually for programs in any other language.

The window in Figure 5 also shows the set of metrics which must be collected in this task (time devoted to the task and affected function-points). It also shows the error origin and cause, according to Briand *et al*. [30]. We have doubted the accuracy of using function-points for estimating the effort of maintenance (overall in *urgent corrective*), but the organization we work with uses function-points as a standard estimation technique.

3.1.2. Task UC2: making corrective actions

Figure 6 shows the window for the second *urgent corrective* task. In this task, the list of modified software elements must be provided, as well as the indicated metrics. For the first task, there is a component of MANTOOL, the 'Measurer' which automatically detects the software elements whose values have changed from the last intervention.

The Measurer (Figure 7) is launched by pressing the *Update elements* button of Figure 6. It estimates some metrics for all the routines of the application corresponding to the MR and updates the database

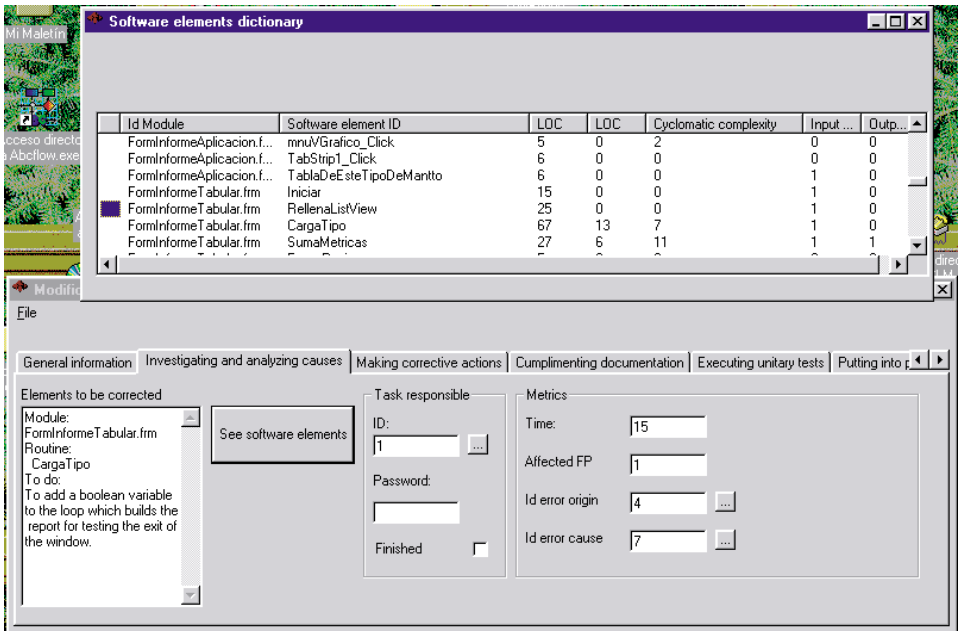

Figure 5. Tab for the first *urgent corrective* task and the software elements dictionary.

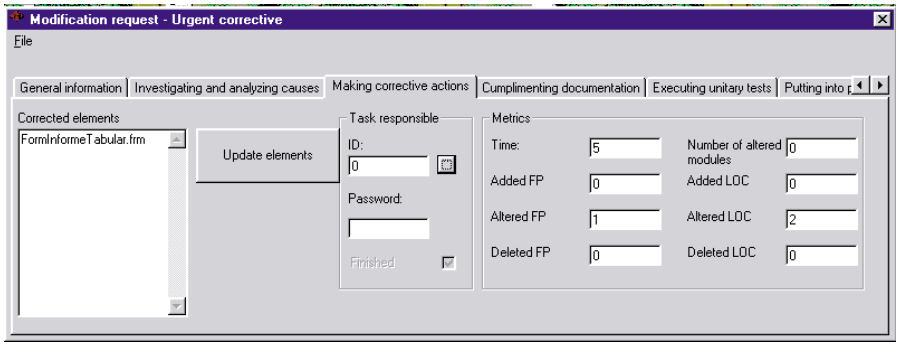

Figure 6. Tab for the second *urgent corrective* task.

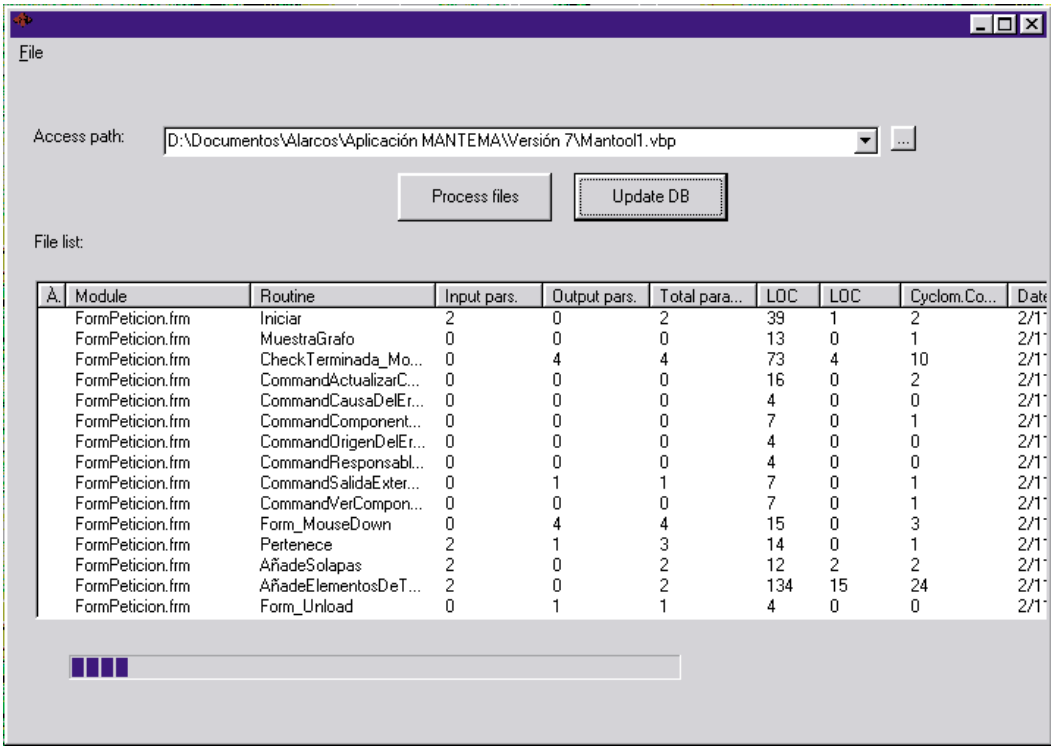

Figure 7. The 'Measurer', updating the database with metrics of modified elements.

with the metrics of all the changed routines. The Measurer results are very useful for detecting changed modules which are automatically passed to the *Corrected elements* text box in Figure 6.

3.1.3. Task UC3: filling in documentation

In this task, a document describing the executed corrective actions must be filled in, as the output of this task. The document corresponding to our MR is shown in Figure 8. This document is saved into the database, and may be consulted at any moment. Also the rest of information on this iteration must be filled in to continue with the following task.

3.1.4. Tasks UC4 (executing unitary tests) and UC5 (putting into production)

The support offered by MANTOOL for these tasks is very similar to the previous ones and the appropriate metrics and the corresponding documents must be filled in.

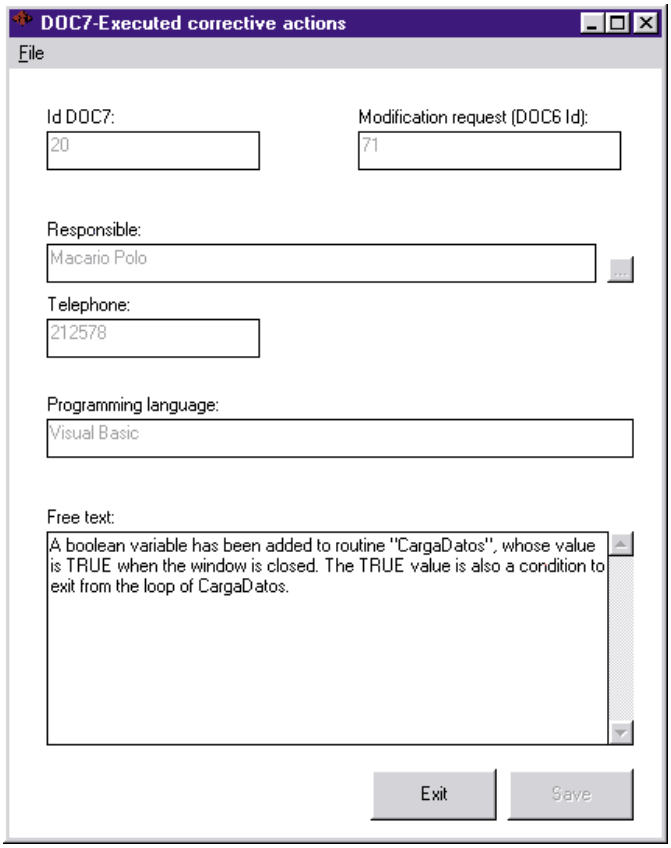

Figure 8. The Executed corrective actions document, once it has been filled in.

3.1.5. Tasks UC6 (documenting correction) and UC7 (saving intervention)

The execution of these two tasks is mandatory when MANTEMA is being applied without automatic support. Note that, in Table I, the input for task UC6 is the complete documentation generated during the intervention, which is grouped into a Full document, which is in turn passed as input to task UC7 for its definitive saving. Obviously, when MANTOOL is used, these two tasks are executed along with the intervention, since MANTOOL keeps all the generated documentation in the database.

3.2. Characteristics of MANTOOL

All the information stored in the MANTOOL database allows us to do several kinds of studies and reports. In this section we will see some of them.

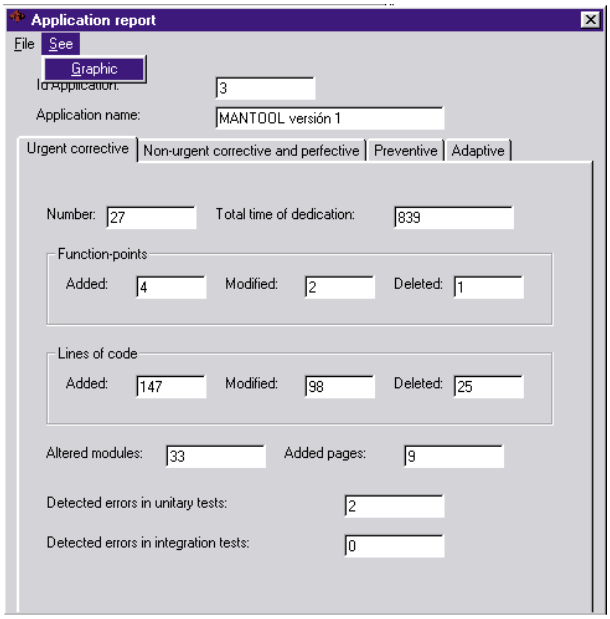

Figure 9. Report about an application.

3.2.1. *Reports of application*

These reports show a summary of the different data saved for every application in every type of maintenance. These include: total time dedicated, total number of added lines of code, total number of errors detected during tests etc. Figure 9 shows one of these reports whose data can also be seen in a graph showing the distribution of the number of modification requests of every type.

3.2.2. *Reports: maintenance type*

These reports consist of a table such as that shown in Figure 10. It collects all data about maintenance requests related to a determined type of maintenance. These tables can be saved as ASCII files in order to be able to use them with any other utility programs. This example illustrates a report of *non-urgent corrective* and *perfective*.

3.2.3. *Reports of dedication of personnel*

These reports show the time devoted to maintenance tasks for all the people in the maintenance organization.

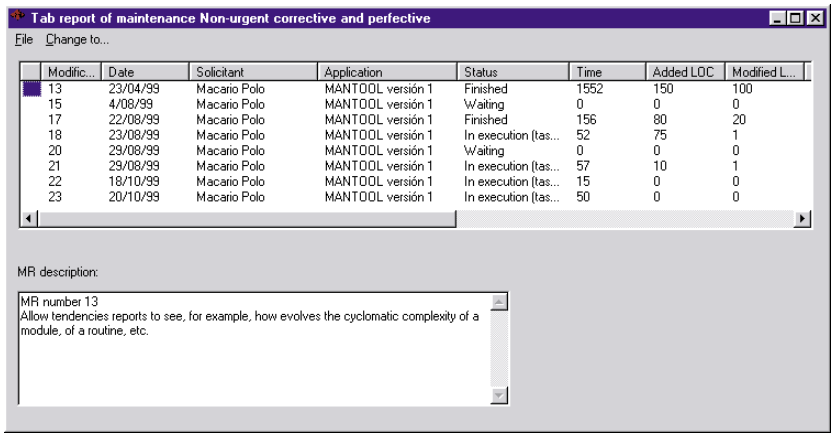

Figure 10. Tabular report of *non-urgent corrective* and *perfective* maintenance.

3.2.4. Reports of deviations

These reports show the difference between the time initially scheduled for executing a MR and the actual time.

3.2.5. Reports of tendency

Thanks to the keeping of records of the product metrics for all interventions, we can see the evolution of the different metrics both of modules and of routines, with the possibility of plotting these data to a graphic. Figure 11, for example, shows the evolution of the routine 'ActualizarAplicacion', providing the evolution of the following: number of input and output parameters of the routines; number of lines of code; number of commented lines of code; cyclomatic complexity; and days from the last change.

This kind of report is interesting for checking the commitment of the preventive maintenance that the maintenance organization could have acquired during the establishment of the outsourcing relationship (e.g. the reduction of the mean cyclomatic complexity).

3.3. Benefits of using MANTOOL

The use of MANTOOL (or of any other tool which helps in the control of software processes) is useful and beneficial because it allows the following to be realized:

3.3.1. Control efforts

A MR implies the realization of a set of tasks which require an effort in time (see sixth column of Figure 10). The analysis of executed tasks and of the time dedicated to each provides valuable information in order to achieve the following:

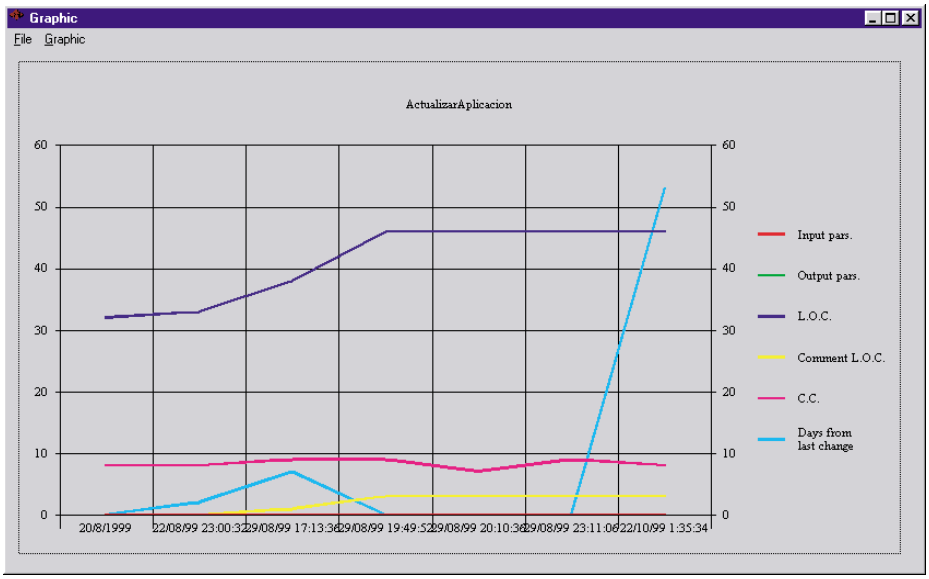

Figure 11. Report of tendency of routine 'ActualizarAplicacion'.

- *Redimensioning of work teams*: tasks which consume many resources in certain work teams can cause the need to increase or modify the team.
- *Supervision of the process quality*: if a high percentage of resources are employed in management tasks, it is maybe the moment to review some procedures which perhaps are too bureaucratic.

3.3.2. *Control of personnel*

Every human resource can charge time to every one of the available tasks. This facilitates the construction of work records. It also helps the staff to find problems in the team (e.g. absence of people with a determined profile, necessity of training in some areas etc.).

3.3.3. *Control of documentation*

Documentation related with every application and intervention is recorded in the database and it is easily available to whoever wishes to consult it. This is a small, but effective, way of helping with configuration management.

3.3.4. *Assessment of future projects*

Since with MANTOOL we can see the costs of maintenance in every application (and these ones are inventoried in the 'software elements dictionary'), MANTOOL can be easily used for making certain

types of statistical analysis with the information available in the database, in order to estimate the cost of future interventions.

3.3.5. Control of the service level fulfilment

When there is an outsourcing relationship, different service level indicators are contracted between the customer organization (the one which needs the maintenance service) and the provider organization at the beginning of the outsourcing relationship. Several indicators can be defined in order to assure the quality standards of the service, for example:

- response times for the interventions of the different types of maintenance; and
- commitment of quality product improvement (for example: to have decreased in certain degree the cyclomatic complexity of applications at the end of the contract).

MANTOOL allows us to keep under control these and other possible indicators.

4. EXPERIENCE REPORTS

The qualitative research method Action Research [32] was used during the construction of some parts of MANTEMA (i.e. process model, definition of deliverables, proposal of one of the techniques and refinement of MANTOOL), whereas other kinds of validation and verification methods were used for metrics and some other techniques. Qualitative methods in general have shown their convenience as good mechanisms for research in Software Engineering [33], and Action Research [34] was particularly acceptable.

Action Research is a cyclic process in which all the actors involved in the research participate. Padak and Padak [31] identify four actors in this method: the *researcher* (the one who carries out research, and which in this case has been constituted by our group in the University); the *researched object* (the problem to be resolved and which here is the Maintenance Process); the *critical reference group* (which is the group who has the problem that the research is intended to resolve and which participates in the researching process—in this case Atos ODS); and the *researched for* (i.e. those who might benefit from better information about the situation, although there is no direct intervention in the researching process, the customers of Atos ODS could be in this group—in this case).

From this scheme, we have followed the usual way for doing this kind of 'active research', as Padak and Padak [31] established (see Figure 12). Initially there were some joint meetings between the researcher and Atos ODS which served to define the problems of maintenance. The context in which the critical reference group operated was explained and lines of approach that could be followed to advance towards the solution were drawn up. After this, there has always been continuous feedback, whose origin was in the different proposals made by the researcher and Atos ODS. In their role of critical reference group, results were reformed about these proposals. In the beginning, as the result of reading and study and then after several researcher cycles (with MANTEMA at an advanced stage of refinement), quantitative results were obtained from the application to the following four real cases of one of the bigger Spanish banking corporations. These were implemented in Cobol, Cobol-DB2, Cobol-CICS and Cobol-CICS-DB2.

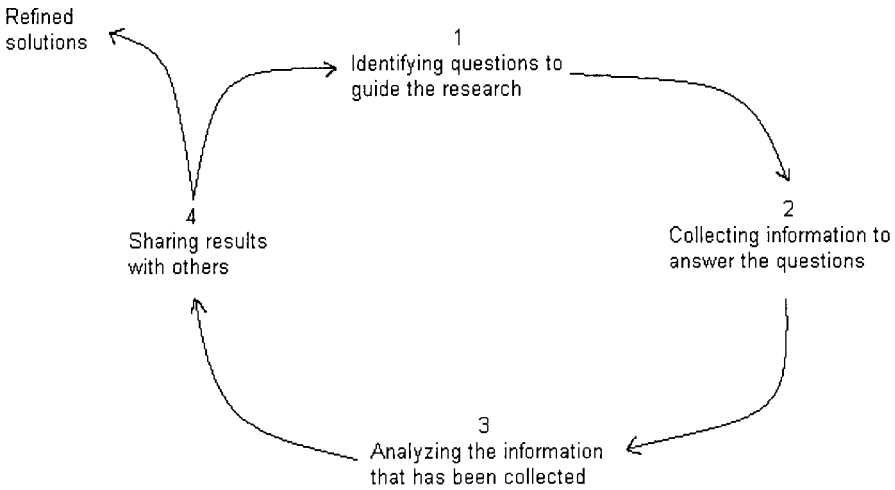

Figure 12. Cyclic process of Action Research.

- *Tax collection*. This application has 135 programs with 103 331 LOC and its main functions are: tax collection for the Ministry of Finance, self-government regions and city councils; information collection about tax refunds to send to the Ministry of Finance; and seizures of accounts due to orders of the Ministry of Finance.
- *Receipt domiciliations*. This application has 157 281 LOC in 196 programs. It manages the payment of the charges which arrive through magnetic support or through electronic interchange with other entities.
- *Transfers*. This application has 308 programs and 247 156 LOC. It manages transfers between accounts of the same or different entities, both periodic and unitary ones.
- *Proactive risk*. This recent project takes advantage of widely available information on the clientele in order to automate the credit concession. It was developed in 1994 and 1995 and runs on an IBM 9000-982 with MVS-TSO. It has 576 programs, 235 with access to DB2 and 94 on-line transactions, which execute 227 different programs.

MANTEMA (with MANTOOL as one of its main components) continues to be used in these projects, and it has been adopted in many others. As we stated in Section 1, it is quite difficult to apply the methodology with no automatic support. The application of Action Research during the building of MANTEMA has allowed the current implementation of MANTOOL to incorporate all the characteristics desired by its users. In all the study cases, all the defined types of maintenance have been successfully applied. Setting aside the specific benefits of MANTOOL (see Section 3.3), some of the benefits of the joint use of MANTEMA and MANTOOL are: the ease of interchanging people among projects (thanks to the definition of roles and team structure); the full definition and updating of all the project documentation; the full detailed specification of all the input and output elements of every task;

and the full and immediate access to all the information of any present or past project, activity, task, modification request, people and team. Moreover, the possession of the methodology (once more, with the integrated tool) has been a key factor in the recent concessions of important maintenance contracts to Atos ODS and in the ISO 9000 certification process of its maintenance service.

5. CONCLUSIONS AND FUTURE WORK

In this paper we have briefly presented MANTEMA, a methodology for managing the software maintenance process and MANTOOL, a tool for controlling and managing the software maintenance process according to this methodology.

This tool is useful in order to maintain under control the state of every past or present project, activity, task, modification request, person and team. It allows engineers to maintain under control the whole maintenance process.

Many 'perfective modification requests' are in the mind of MANTOOL and MANTEMA authors. Setting aside new graphics and reports, those which concern us most are as follows:

- Full support of MANTEMA, allowing the addition of the initial and final set of activities and tasks. However, we have developed recently SREM (Single Resource Estimating for Maintenance), a vertical tool which will be shortly integrated in MANTOOL. SREM is used for estimating resources for non-plannable maintenance according to the technique described by Polo *et al*. [29].
- Addition of complements for estimating costs of future projects with CREM (Complex Resource Estimating for Maintenance), a tool that we are now developing.
- Addition of a complement to maintain the control of the evolution of metrics for databases (Calero *et al*. [35]).
- Construction of several modules that act as interfaces between MANTOOL and some commercial tools for software measurement. With this, the possibility of using such tools as components of MANTOOL will allow us to estimate metrics for applications in many programming languages. This implies the definition of several interfaces and, in a future, maybe the rebuilding of MANTOOL with Workflow tools and techniques.

ACKNOWLEDGEMENTS

This work is partially supported by the projects: MANTIS (CICYT/EU, 1FD97-1608) and MPM: Mejora del Proceso de Mantenimiento (Ministerio de Industria y Energía, Iniciativa ATYCA, TA15/1999).

REFERENCES

1. Lear AC. Cobol programmers could be key to new IT. *Computer* 2000; **33**(4):1–19.
2. Brereton P, Budgen D, Hamilton G. Hypertext: the next maintenance mountain. *Computer* 1999; **31**(12):49–55.
3. Taschwer M, Rauner-Reithmayer D, y Mittermeir R. Generating objects from C code. *Proceedings Third European Conference on Software Maintenance and Reengineering*. IEEE Computer Society: Los Alamitos CA, 1999; 91–100.

4. Jahnke JH, Wadsack J. Integration of analysis and redesign activities in information system reengineering. *Proceedings Third European Conference on Software Maintenance and Reengineering*. IEEE Computer Society: Los Alamitos CA, 1999; 160–168.
5. Singer J. Practices of software maintenance. *Proceedings International Conference on Software Maintenance*. IEEE Computer Society Press: Los Alamitos CA, 1998; 139–145.
6. van Bon J. Sourcing. *World Class IT Service Management*, van Bon (eds.). ten Hagen & Stam Publishers: The Hague, 2000.
7. Basili V, Briand L, Condon S, Kim Y, Melo W, Valett JD. Understanding and predicting the process of software maintenance releases. *Proceedings International Conference on Software Engineering*. IEEE Computer Society Press: Los Alamitos CA, 1996; 227–231.
8. Schach SR, Tomer A. A maintenance-oriented approach to software construction. *Journal on Software Maintenance* 2000; **12**(1):25–45.
9. Alexander LC, Davis AA. Criteria for selecting software process models. *Proceedings Computer Software and Applications Conference (COMPSAC'91)*. IEEE Computer Society Press: Los Alamitos CA, 1991; 521–528.
10. Gillies AC. *Software Quality: Theory and Management*. Chapman & Hall: London, 1992.
11. Pigoski TM. *Practical Software Maintenance*. John Wiley & Sons: New York, 1997.
12. Ruiz F, Piattini M, Polo M, Calero C. Audit of software maintenance process. *Auditing Information Systems*, Piattini (ed.). Idea Group Publishing: Hershey, PA.
13. Quang PT. *Réussir la Mise en Place du Génie Logiciel et de l'Assurance Qualité*. Eyrolles: Paris, France, 1993.
14. Queen's University. CASE tool index. http://www.qucis.queensu.ca/Software-Engineering/tools.html [10 July 2000].
15. Mazza C, Fairclough J, Melton B, de Pablo D, Scheffer A, Stevens R. *Software Engineering Guides*. Prentice Hall: UK, 1996.
16. Polo M, Piattini M, Ruiz F, Calero C. MANTEMA: A complete rigorous methodology for supporting maintenance based on the ISO/IEC 12207 standard. *Proceedings Third European Conference on Software Maintenance and Reengineering*. IEEE Computer Society Press: Los Alamitos, CA, 1999a; 178–181.
17. International Standard Organization/International Electrotechnical Committee. Information Technology-Software Life Cycle Processes; ISO/IEC 12207, Geneva, Switzerland 1995.
18. Graham I, Henderson-Sellers B, Younessi H. *The OPEN Process Specification*. ACM Press and Addison-Wesley: Essex, UK, 1997.
19. Rao HR, Nam K, Chaudhury A. Information systems outsourcing. *Communications of the ACM* 1996; **39**(7):27–28.
20. Hoffman T. Users say move quickly when outsourcing your personnel. *Computer World* 17 March 1997; 77. http://www.computerworld.com/cwi/story/0,1199,NAV47_ST02167,00.htm [21 February 2001].
21. Klepper R, Jones WO. *Outsourcing Information Technology, Systems and Services*. Prentice-Hall: New Jersey, 1998.
22. Brower JM. Outsourcing and privatizing information technology. *Crosstalk, The Journal of Defense Software Engineering* 1999; **12**(1):28–30.
23. Polo M, Piattini M, Ruiz F, Calero C. Roles in the maintenance process. *ACM Software Engineering Notes* 1999; **24**(4):84–86.
24. Sneed HM, Foshag O. Measuring legacy database structures. *Proceedings of the First European Software Measurement Conference*, Coombes H, Hooft van Huysduynen M, Peeters B (eds.). Technological Institute: Antwerp, Belgium, 1998; 199–211.
25. Genero M, Jiménez L, Piattini M. Measuring the quality of entity relationship diagrams. *Entity Relationship (ER'2000)* (*Lecture Notes in Computer Science*, vol. 1920), Laender A, Liddle SW, Storey VC (eds.). Springer, 2000; 513–526.
26. Piattini M, Calero C, Polo M, Ruiz F. Towards a metric suite for relational databases complexity. *Proceedings 5th World Conference on Integrated Design and Process Technology*, Society for Design and Process Science, CD-ROM, 2000.
27. Piattini M, Calero C, Sahraoui HA, Luonis H. An empirical study with object-relational databases metrics. *4th International Workshop on Quantitative Approaches in Object-Oriented Software Engineering (in ECOOP'2000)*, Centre de recherche informatique de Montréal, Montréal, Quebec, 2000; 39–46.
28. Díaz Ó, Piattini M, Calero C. Measuring triggering interaction complexity on active databases. *Information Systems Journal* 2000. To be published.
29. Polo M, Piattini M, Ruiz F. Planning the non-plannable maintenance. *Project Control: The Human Factor, Proceedings of the combined 11th European Software Control and Metrics Conference and the 3rd SCOPE Conference on Software Product Quality (ESCOM-SCOPE 2000)*. Shaker Publishing: Maastricht, The Netherlands, 2000; 49–57.
30. Briand L, Kim Y, Melo W, Seaman C, Basili VR. Q-MOPP: Qualitative evaluation of maintenance organizations, processes and products. *Journal of Software Maintenance* 1998; **10**(4):249–278.
31. Padak N, Padak G. Guidelines for planning action research projects. http://archon.educ.kent.edu/Oasis/Pubs/0200-08.html [10 July 1994].
32. McTaggart R. Principles of participatory action research. *Adult Education Quarterly* 1991; **41**(3):170.

33. Seaman CB. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering* 1999; **25**(4):557–572.
34. Avison D, Lan F, Myers M, Nielsen A. Action research. *Communications of the ACM* 1999; **42**(1):94–97.
35. Calero C, Pascual C, Serrano MA, Piattini M. Measuring Oracle database schemas. *Computers and Computational Engineering and Control*, World Scientific and Engineering Society Press: Athens, Greece, 1999; 237–243.

AUTHORS' BIOGRAPHIES

Macario Polo is full time Professor at the Department of Computer Science in the University of Castilla-La Mancha in Ciudad Real, Spain. He received a MS degree in Computer Science from the University of Seville (Spain) and a PhD degree with European Accreditation in Computer Science from the University of Castilla-La Mancha. His current research areas include software maintenance, process improvement and the development of tools to automatize software processes. He has published several technical articles and papers on these themes, as well as two novels in Spanish. He can be contacted at mpolo@inf-cr.uclm.es.

Mario Piattini received a MSc and a PhD in Computer Science at the Politechnical University of Madrid. He is a Certified Information System Auditor registered by ISACA (Information System Audit and Control Association) and an Associate Professor at the Department of Computer Science in the University of Castilla-La Mancha in Ciudad Real, Spain. He is the author of several books and papers on databases, software engineering and information systems. He leads the Alarcos research group of the Department of Computer Science at the University of Castilla-La Mancha in Ciudad Real, Spain. His research interests are: advanced databases design, database quality, software metrics, object oriented metrics and software maintenance. His E-mail address is mpiattin@inf-cr.uclm.es

Francisco Ruiz is full time Professor of the Department of Computer Science at University of Castilla-La Mancha (UCLM) in Ciudad Real, Spain. He was Dean of the Faculty of Computer Science (Escuela Superior de Informaática) for seven years until February 2000. Previously, he was Computer Services Director at the university and he has also worked in private companies as an analyst-programmer and project manager of information systems. His current research interests include: software maintenance, enhancement of software processes, metrics for object-oriented databases and methodologies for planning and managing software projects. In the past, other work topics have been: GIS (geographical information systems), educational software systems and deductive databases. He has written five books on these topics and he published 30 papers in international congresses, conference and magazines. He belongs to various scientific and professional associations (ACM, IEEE-CS, ATI, AEC, AENOR, ISO JTC1/SC7 and ACTA). His E-mail address is fruiz@inf-cr.uclm.es